yang_seonhyeHW22

```r
library(mfp)
```

```
## Loading required package: survival
```

```r
data("bodyfat")
```

Question 0

```r
# Doing a 50-50 Training-Testing split
set.seed(10)
sampleRows <- sample.int(nrow(bodyfat), floor(.5 * nrow(bodyfat)))
training <- bodyfat[sampleRows, ]
testing  <- bodyfat[-sampleRows, ]
```

Question 1

Part A: Ridge Regression

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```r
trainingY <- as.matrix(training["brozek"])
trainingX <- as.matrix(subset(training, select=-c(brozek, siri, case, density)))
testingY <- as.matrix(testing["brozek"])
testingX <- as.matrix(subset(testing, select=-c(brozek, siri, case, density)))
```

```r
ridge_mod <- glmnet(trainingX, trainingY, alpha = 0)
ridge_cv  <- cv.glmnet(trainingX, trainingY, alpha = 0)
ridgeTunedValue <- ridge_cv$lambda.min
ridgeTunedValue
```

```
## [1] 0.7335467
```

Part B: Lasso Regression

```r
lasso_mod <- glmnet(trainingX, trainingY, alpha = 1)
lasso_cv  <- cv.glmnet(trainingX, trainingY, alpha = 1)
lassoTunedValue <- lasso_cv$lambda.min
lassoTunedValue
```

```
## [1] 0.3102347
```

For both cases, we used the same methodology to find the best tuning parameter, λ. This method was cross-validation. This is done using the default of 10 folds, and by applying this method, we can find find the minimum λ for our training set. This should then give us the most tuned parameter, ideally giving us the smallest MSE possible.

Question 2

```
ridge_predict <- predict(ridge_mod, s=ridgeTunedValue, newx = testingX)
ridgeMSE <- mean((ridge_predict - testingY)^2)
ridgeMSE
```

```
## [1] 19.17209
```

```
lasso_predict <- predict(lasso_mod, s=lassoTunedValue, newx = testingX)
lassoMSE <- mean((lasso_predict - testingY)^2)
lassoMSE
```

```
## [1] 18.68741
```

Question 3

Based on the results of Question 2, we can see that the lasso model had a lower MSE of 18.4, compared to the 19.1 of the ridge regression. We can see from the following R snippets what variables were kept by our models:

```
coefRidge <- predict(ridge_mod, type = "coefficients", s = ridgeTunedValue)[1:14, ]
coefRidge[coefRidge != 0]
```

```
##   (Intercept)           age        weight        height          neck
## -1.701277e+01  1.218501e-01  1.409677e-04 -1.188252e-01 -3.187374e-01
##         chest       abdomen           hip         thigh          knee
##  1.713460e-01  3.762046e-01 -1.058876e-02  2.577528e-01  4.788571e-02
##         ankle        biceps       forearm         wrist
## -5.176802e-02  1.263285e-01  2.882778e-01 -1.568240e+00
```

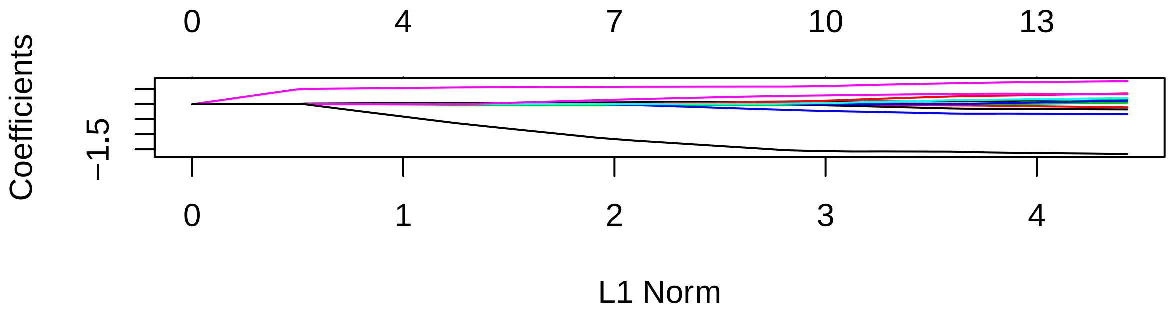
```
coefLasso <- predict(lasso_mod, type = "coefficients", s = lassoTunedValue)[1:14, ]
coefLasso[coefLasso != 0]
```

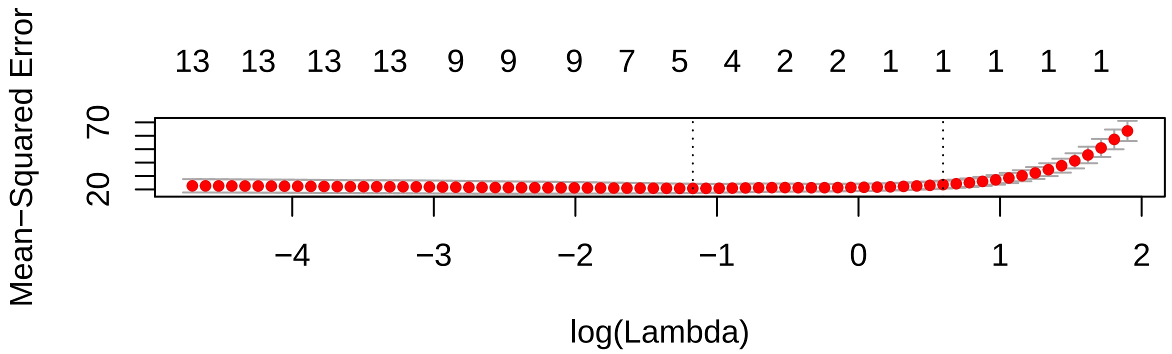
```
##  (Intercept)          age       height      abdomen      forearm
## -20.34643004   0.05128462  -0.02460889   0.57082960   0.07031808
##        wrist
##  -0.89398534
```

This shows us that the ridge regression does not perform variable selection, while the lasso regression has dropped many variables and kept only age, height, abdomen, forearm, wrist. This explains why the lasso regression may have a smaller MSE, as it is less susceptible to leverage from unnecessary variables.

Looking at the various plots for the lasso model:

```
par(mfrow=c(2,1))
plot(lasso_mod)
plot(lasso_cv)
```





Seeing these plots, we can notice where both the best lambda is picked, and how the lasso model will zero out the coefficients of variables based on the lambda you pick, producing a better model than the ridge, as measured by MSE.